Attorney Docket No. 50325-0543

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

#19
O. Cato
4-15-04

In re application of:

Alan NEWMAN et al.

Confirmation No. 2877

Group Art Unit No.: 2124

Serial No.: 09/407,531

Examiner: John Q. CHAVIS

Filed: September 28, 1999

For: A METHOD AND SYSTEM FOR A SOFTWARE RELEASE PROCESS

MS Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed December 23, 2003.

I. REAL PARTY IN INTEREST

Cisco Systems, Inc. is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-3, 8, 9, 14, 15, and 19-31 are pending in this application, have been finally rejected, and are the subject of this appeal.

IV. STATUS OF AMENDMENTS

No amendments were filed after the Final Office Action mailed on July 2, 2003.

V. SUMMARY OF THE INVENTION

New and different computing devices are continuously being developed and offered to the public. Different computing devices may have different hardware configurations and/or different operating system configurations. A computing device having a particular type of hardware and/or operating system configuration may be called a "platform". Some platforms are incompatible. As a result, a software application that has been designed to execute on one platform might not execute on another platform. In order to accommodate different platforms, different platform-specific versions of the same software application often must be created.

Users of a software application may discover that the software application lacks some desirable functionality. The users may convey their desires for new functionality, or in other words, new features, to the software application developer. In response, the software application developer may create a new version of the software application that provides new features that were not present in prior versions of the software application. During the creation of the new version, the same new features may be incorporated into multiple different existing platform-specific versions of the software application. When

new features do not vary between platform-specific versions of the software application, the new features may be said to be "platform-independent".

To remain competitive with other software application developers in the same market, a software application developer may need to release new versions of its software application to the public at frequent time intervals. Prior to releasing the new version, the software application developer will want to test the new version to discover any errors and repair those errors. A traditional approach to software application development, testing, and release is illustrated in Figure 3 of the specification. Line 301 represents a main development stream of a software application. The same single version or code base of the software application's source code is typically used in main development stream 301. As time progresses, changes may be applied to the software application's source code within main development stream 301. These changes may be related to the addition of new features to the software application, for example.

At some point in time 305, a separate "branch" 307 of development is sprouted off of the main development stream 301 to facilitate testing preparatory to a release of a new version to the public. A copy of the software application's source code, separate from the source code for the main development stream, is created for branch 307. As time progresses beyond point 305, new features may be added to the source code associated with the main development stream 301 (the "main code"), but new features may not be added to the source code associated with branch 307 (the "branch code"). The branch code is "frozen" relative to the addition of new features. The only changes that are made to the branch code are repairs to errors that are discovered during the testing of the branch code after point 305. Such repairs may also be applied to the main

code. At some later point in time 309, when the branch code has passed testing, the branch code is compiled and released as a new version of the software application.

Formerly, in order to distribute human labor according to expertise, the development of source code to support new platform-independent features ("feature code") was separated from the development of source code to support new platforms ("platform code"). The development of feature code was assigned to one development group, and the development of platform code was assigned to another development group. In this manner, the feature code and the platform code could be developed in parallel.

Figure 4 of the specification illustrates this separated feature/platform development process. A platform code development stream 403 is shown separate from a feature code development stream 405. At some point in time, represented by points 417 and 423 in development streams 403 and 405, respectively, separate branches 419 and 425 are sprouted off of development streams 403 and 405, respectively. As time progresses, repairs to the platform code and repairs to the feature code are made, separately, within branches 419 and 425, respectively.

The problem with the separated feature/platform development process becomes most acute when the feature code is merged with the platform code in preparation for a release. During the separate testing of the platform code and feature code, repairs to the platform code often should be made to the feature code as well, and vice-versa. However, because the development groups for the platform code and feature code are separate, repairs to the platform code often are not made to the feature code, and vice-versa. The merging of the feature code with the platform code can introduce new and

previously undiscovered errors. To complicate matters further, the merging of particular feature code with platform code for one platform can introduce errors that are different than errors introduced by the merging of the particular feature code with platform code for another, different platform. All too often, insufficient time remains, prior to a scheduled release, to test and repair such errors.

The invention recited in Claims 1-3, 8, 9, 14, 15, and 19-31 addresses the problem of how to develop new versions of a software application so that the new versions can be released, relatively error-free, at scheduled times while providing support in those new versions for both new platforms and new platform-independent features. According to one aspect of the invention, this problem generally is addressed by requiring each development group in a plurality of separate development groups to pre-test and commit its software modules into a common development branch early enough to allow sufficient time for all of those software modules to be tested in combination, within the common development branch, prior to a scheduled release. The requirement is imposed on each development group regardless of whether the development group's software modules are designed to support new platforms or new platform-independent features. Thus, the common development branch comprises pre-tested modules for new platforms as well as pre-tested modules for new platform-independent features.

According to another aspect of the invention, this problem generally is addressed by requiring each of the development groups that develops software modules that support new platform-independent features to select, from among those features, a group of features that can be pre-tested and committed into the common development branch early

enough to allow sufficient time, prior to the scheduled release time, for the aforementioned combined testing.

Figure 6 of the specification illustrates a development system, according to an embodiment of the invention, in which software modules from a plurality of development groups, or in other words, "business units," are pre-tested by those business units in separate "pre-integration" branches before being committed into a common branch for combined testing prior to release. Separate pre-integration branches 627, 633, 639, and 645 are shown. Each of these pre-integration branches corresponds to a separate business unit. Each business unit is responsible for the development of one or more new features for a software application.

At a first point in time, each business unit is instructed to select, from among the new features for which that business is responsible, a set of features that can be tested, separately and in combination with each other, prior to a second specified point in time that is based on the scheduled release time. The features may be selected in a manner such that the set of features includes a maximal number of features, or such that the set of features includes features having a maximal combined value, in view of the time constraint. Once a business unit has selected a set of features, the business unit pre-tests the software modules that implement those features separately and in combination with each other. This pre-testing is performed in the business unit's corresponding pre-integration branch.

After each business unit has pre-tested the software modules in its pre-integration branch, each business unit commits its pre-tested software modules into a common branch 611 no later than the specified second time. Within common branch 611, the pre-

tested software modules from the plurality of business units are tested in combination with each other. After the software modules in the common branch have been tested in combination, those software modules are provided to a development branch 603 in time to be released to the public at the scheduled release time.

VI. ISSUES

The sole issue is whether Claims 1-3, 8, 9, 14, 15, and 19-31 are anticipated under 35 U.S.C. § 102(e) by Hopwood et al., U.S. Patent No. 6,223,343 B1 ("Hopwood").

VII. GROUPING OF CLAIMS

Claims 1-3, 8, 9, 14, 15, and 19-31 do not fall or stand together, and the following groupings are asserted:

GROUP 1: Claims 1, 2, and 8;

GROUP 2: Claims 3, 9, and 19-31;

GROUP 3: Claims 14 and 15.

VIII. ARGUMENTS

A. Introduction

In the Final Office Action dated July 2, 2003, Claims 1-3, 8, 9, 14, 15, and 19-31 were rejected under 35 U.S.C. § 102(e) as being anticipated by Hopwood. Appellants respectfully submit that Claims 1-3, 8, 9, 14, 15, and 19-31 are patentable over Hopwood for at least the reasons provided hereinafter.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference."

Verdegaal Bros. v. Union Oil Co. of California, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed Cir. 1987). Furthermore, "[t]he identical invention must be shown in as complete detail as is contained in the claim." *Richardson v. Suzuki Motor Co.,* 868 F.2d 1226, 1263, 9 UPSQ2d 1566 (Fed. Cir. 1989). With respect to the present application, Appellants respectfully submit that Hopwood does not in any way teach or suggest all the features and limitations of Claims 1-3, 8, 9, 14, 15, and 19-31.

B. Claims 1, 2, and 8 (GROUP 1) Are Patentable under 35 U.S.C. § 102(e) over Hopwood Because Hopwood Does Not in Any Way Teach or Suggest the Elements of these Claims

Claim 1 addresses the problem of how to develop new versions of a software application so that the new versions can be released, relatively error-free, at scheduled times while providing support in those new versions for both new platforms and new platform-independent features. According to the approach recited in Claim 1, an "early development" branch is provided. The "early development" branch is designated for the incorporation of one or more software modules that provide support for new features and platforms. Pre-tested software modules are received from a plurality of integration units. Each of the pre-tested software modules comprises one or more new features or supports one or more new platforms. The pre-tested software modules, for both new features and new platforms, are committed into the "early development" branch. Using the "early development" branch, a new "early development" release is generated. The new release contains the pre-tested software modules.

However, Hopwood does not in any way describe the approach recited in Claim 1, because Claim 1 includes at least one limitation that is not in any way described by Hopwood.

For example, Claim 1 requires "providing an early development branch . . . that is designated for incorporation of one or more software modules providing support for new features and platforms." Thus, Claim 1 requires that a development branch be **designated** for incorporating software modules that provide support for **both new features and new platforms**.

In the Final Office Action, the Examiner states, in regard to this limitation, "See the developer branch (items 90, 96, and 104) in fig. 6, col. 6 lines 29-32, and col. 2 lines 9-10, which indicates that new features are tested. See also col. 20 lines 25-33." Col. 6, lines 29-32 read: "Thus, the work station developer is able to group together one or more elements to be downloaded into the destination systems." Col. 2, lines 9-10 read: "Developer platform 30 includes development tools for creating and testing new development programs." Neither of these passages says anything about a branch that is **designated** for incorporating software modules that provide support for **both new features and new platforms**. Appellants do not contend that the creating and testing of new programs is novel.

Col. 20, lines 25-33 describe that a branch is a "line of development," but fail to describe any branch that is **designated** for incorporating software modules that provide support for **both new features and new platforms**. Appellants do not contend that the concept of a development branch is novel.

The Examiner also alleges that Hopwood's col. 14, lines 30-41 describes software modules that support new platforms. This section of Hopwood indicates that a Revision Management System (RMS) may perform revision management services on different development systems, such as IBM and HP host machines. Thus, Hopwood describes that developers who use different systems can access the RMS through a common interface that does not vary from system to system, thereby bringing all developers "under the same interface" and providing them with the same revision management tools.

This does not mean that the RMS manages changes to software modules that provide support for new platforms. Even if one developer may use the RMS on an IBM host machine and another developer may use the RMS on an HP host machine, this does not mean that the targets of the revisions managed by the RMS systems necessarily provide support for new platforms. The Examiner seems to have confused Hopwood's disclosure of the ability of the RMS to execute on different platforms with a supposed but non-existent disclosure that the RMS manages revisions to software modules that support different platforms.

The Examiner also alleges that Hopwood's col. 2, lines 6-8 describes software modules that support new platforms. This section of Hopwood indicates that a system comprises a support platform that may be used to load new programs and operating systems, to evaluate the use of those programs and operating systems on the system.

This does not mean that the elements managed by the RMS, which executes on the system, provide support for new platforms. Even if the system on which the RMS executes may include a "platform" on which a new operating system may be evaluated prior to that new operating system being used in the remainder of the system, this does

not mean that the targets of the revisions managed by the RMS necessarily provide support for new platforms. The Examiner seems to have confused the ability of the system on which the RMS executes to evaluate new operating systems on a separate platform, as described in Hopwood, with a supposed ability of the RMS to manage revisions to software modules that support different platforms.

Therefore, based on the foregoing, Appellants respectfully submit that Claim 1 recites at least one limitation that Hopwood does not in any way teach or suggest. Specifically, Hopwood fails to teach or suggest the limitation of "providing an early development branch . . . that is **designated** for incorporation of one or more software modules providing support for **new features and platforms**." Therefore, Claim 1 is patentable over Hopwood.

Claim 2 depends from Claim 1 and includes all of the elements of Claim 1. Therefore, Appellants respectfully submit that Claim 2 is patentable over Hopwood for at least the reasons set forth herein with respect to Claim 1.

Claim 8 recites a system that comprises mechanisms for performing all of the steps of Claim 1. Therefore, Appellants respectfully submit that Claim 8 is patentable over Hopwood for at least the reasons set forth herein with respect to Claim 1.

C. Claims 3, 9, and 19-31 (GROUP 2) Are Patentable under 35 U.S.C. § 102(e) over Hopwood Because Hopwood Does Not in Any Way Teach or Suggest the Elements of these Claims

In addition to the limitations of Claim 1, Claim 3 additionally requires that the **pre-tested software modules be received at a pre-integration branch** that is separate

from the early development branch. The Examiner alleges that the pre-integration branch is disclosed in the form of Hopwood's sub-repository 106.

However, in col. 15, lines 47-51, Hopwood discloses that sub-repository 106 centralizes predetermined information that is a **subset of repository 100**. In col. 15, lines 42-46, Hopwood discloses that repository 100 centralizes information about the business and that applications which support it, and then parenthetically elaborates that this information is application and business **metadata**. Therefore, because sub-repository 106 is a subset of repository 100, sub-repository 106 also consists of **metadata**.

In the remainder of col. 15, which the Examiner cites in the rejection of Claim 3, Hopwood goes on to state that this metadata is "element specific information" in that it is information **about** elements (i.e., element metadata), and not the elements themselves. For example, Hopwood discloses in the remainder of col. 15 that the RMS may be used to enter "element change information, including what element needs to be issued, who is issuing the element, when does the element need to be issued, and where does the changed element need to go for update purposes and why." There is no disclosure in Hopwood that this metadata comprises **pre-tested software modules** such as those that the pre-integration branch of Claim 3 is required to receive.

Metadata typically is defined as "data about data." The term "pre-tested software modules," on the other hand, ordinarily refers to executable code or source code logic and cannot truly be described as "data about data." Therefore, contrary to the Examiner's allegations, Hopwood's sub-repository 106 is **not** the same as the pre-integration branch of Claim 3.

Therefore, based on the foregoing, Appellants respectfully submit that Claim 3 recites at least one limitation that Hopwood does not in any way teach or suggest. Specifically, Hopwood fails to teach or suggest the limitation of "wherein the pre-tested software modules are received at a pre-integration branch that is separate from the early development branch." Therefore, Claim 3 is patentable over Hopwood.

Claims 9 and 19-31 similarly recite at least one pre-integration branch. The Examiner offers no rationale for the rejection of Claims 9 and 19-31 beyond the rationale offered for the rejection of Claim 3. Therefore, Appellants respectfully submit that Claims 9 and 19-31 are patentable over Hopwood for at least the reasons set forth herein with respect to Claim 3.

D. Claims 14 and 15 (GROUP 3) Are Patentable under 35 U.S.C. § 102(e) over Hopwood Because Hopwood Does Not in Any Way Teach or Suggest the Elements of these Claims

Claim 14 addresses the problem of how to develop new versions of a software application so that the new versions can be released, relatively error-free, at scheduled times while providing maximal support for new features. According to the approach recited in Claim 14, one or more features are selected for inclusion in a new release of a software system code base, such that the quantity of features selected will allow a next scheduled release of the software system code base to be completed at a required time. The selected features are tested in a plurality of business units. The selected features are provided to a pre-integration branch only when testing in the business units is successful. The selected features are tested in the pre-integration branch. The selected features are provided to a development branch only when testing in the business units is successful

and in time to allow the next scheduled release of the software system code base to be completed in the required time.

However, Hopwood does not in any way describe the approach recited in Claim 14, because Claim 14 includes at least one limitation that is not in any way described by Hopwood.

For example, Claim 14 requires "**selecting one or more features** for inclusion in a new release of the software system code base, **wherein a quantity of features selected will allow a next scheduled release of the software system code base to be completed at a required time**." Thus, Claim 14 requires that features be selected for inclusion in a scheduled release based on whether the inclusion of those features will allow the release to be completed at the required time. This is significantly different than an approach in which a release is scheduled to accommodate the readiness of all features that may be included in the release.

In the Final Office Action, the Examiner relies on col. 17, lines 13-37, and col. 3, lines 33-41 of Hopwood to disclose the scheduling of software updates. However, neither of these passages says anything about **selecting a quantity of features that will allow a next scheduled release to be completed at a required time**. Appellants do not contend that the scheduling of a software update is novel.

The Examiner also cites col. 2, lines 3-13 of Hopwood, which disclose a developer platform on which new development programs may be created and tested, and a production platform that distributes tested programs to components of a system. The Examiner asserts that the purpose of testing is inherently to enable a release decision to be made. Appellants do not contend that the testing of a program or the making of a release

decision is novel. However, Hopwood fails to disclose the selection of features, which at the time of selection may be completely untested, based on a scheduled release date. Indeed, the Examiner has not even alleged that Hopwood discloses this selection of features. Instead, what the Examiner has alleged is more like the selection of a release date based on whether all features have been tested.

Therefore, based on the foregoing, Appellants respectfully submit that Claim 14 recites at least one limitation that Hopwood does not in any way teach or suggest. Specifically, Hopwood fails to teach or suggest the limitation of "**selecting one or more features** for inclusion in a new release of the software system code base, **wherein a quantity of features selected will allow a next scheduled release of the software system code base to be completed at a required time**." Therefore, Claim 14 is patentable over Hopwood.

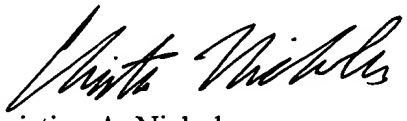
Claim 15 depends from Claim 14 and includes all of the elements of Claim 14. Therefore, Appellants respectfully submit that Claim 15 is patentable over Hopwood for at least the reasons set forth herein with respect to Claim 14.

IX. CONCLUSION AND PRAYER FOR RELIEF

Based on the foregoing, it is respectfully submitted that the rejections of Claims 1-3, 8, 9, 14, 15, and 19-31 under 35 U.S.C. § 102(e) lack the requisite factual and legal bases. Appellants therefore respectfully request that the Honorable Board reverse the rejection of Claims 1-3, 8, 9, 14, 15, and 19-31 under 35 U.S.C. § 102(e) over Hopwood.

Respectfully submitted,

HICKMAN PALERMO TRUONG & BECKER LLP



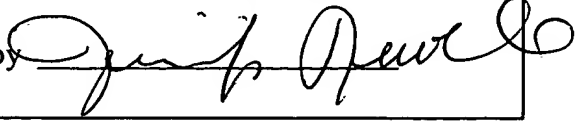
Date: February 13, 2003

Christian A. Nicholes
Registration No. 50,266

1600 Willow Street
San Jose, California 95125-5106
Tel: (408) 414-1224
Fax: (408) 414-1076

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

on 2/13/04 by 

CLAIMS APPENDIX

1. A release control method for providing early deployment releases of a software system, the early deployment releases containing support for new features and platforms, the method comprising the steps of:

a. providing an early development branch of the software system that is designated for incorporation of one or more software modules providing support for new features and platforms;

b. receiving, from a plurality of integration units, a plurality of pre-tested software modules, wherein each of the pre-tested software modules comprises one or more new features or supports one or more new platforms;

c. committing the pre-tested software modules for new features and platforms into the early development branch; and

d. using the early development branch, generating a new early development release containing pre-tested software modules for new features and platforms.

2. The release control method of claim 1 comprising the additional step of repeating steps c and d on a regular recurring basis for a fixed number of cycles.

3. The release control method of claim 1 wherein the pre-tested software module is received at a pre-integration branch that is separate from the early development

branch, and wherein the committing step comprises committing pre-tested software modules for new features and platforms from a pre-integration branch into the early development branch.

8. A system for providing early deployment releases of a software system, the early deployment releases containing support for new features and platforms, comprising:

a. an early development branch of the software system designated for incorporation of one or more software modules providing support for new features and platforms;

b. logic for receiving, from a plurality of integration units, a plurality of pre-tested software modules, wherein each of the pre-tested software modules comprises one or more new features or supports one or more new platforms;

c. logic for committing the pre-tested software modules for new features and platforms into the early development branch;

d. using the early development branch, logic for generating a new early development release containing pre-tested software modules for new features or platforms on a regular recurring basis for a fixed number of cycles; and

e. logic for generating said new early development release containing pre-tested software modules for new features or platforms on a regular recurring basis for a fixed number of cycles.

9. The system of claim 8 wherein the logic for committing comprises logic for committing pre-tested software modules for new features and platforms from a pre-integration branch into the early development branch.

14. A product release method for controlling the release of software system code based on a fixed frequency, the method comprising the steps of:

a. selecting one or more features for inclusion in a new release of the software system code base, wherein a quantity of features selected will allow a next scheduled release of the software system code base to be completed at a required time;

b. testing the quantity of features selected in a plurality of business units;

c. providing the quantity of features selected to a pre-integration branch of the software system code base only when testing in the business units is successful;

d. testing the quantity of features selected in the pre-integration branch;

e. providing the quantity of features selected to a development branch only when testing in the business units is successful and in time to allow the next scheduled release of the software system code base to be completed in the required time.

15. The method of claim 14 comprising the additional steps of:

a. completing testing of a modified software system code base in the development branch which contains the quantity of features selected and tested in the pre-integration branch; and

b. releasing the modified software system code base at the required time.

19. A method as recited in Claim 1, further comprising the steps of:

receiving and testing a plurality of software source code modules that support new features or platforms at a respective plurality of business unit pre-integration branches;

committing one or more of the plurality of software source code modules from the one or more of the business unit pre-integration branches to a central pre-integration branch only when such testing is successful; and

committing the plurality of software source code modules from the central pre-integration branch to the early development branch when all the modules have been committed from the business unit pre-integration branches to the central pre-integration branches.

20. A method as recited in Claim 19, further comprising the step of generating, using the early development branch, a new early development release containing pre-tested source code for new features and platforms only when the plurality of

software source code modules has been committed from the central pre-integration branch to the early development branch.

21. A method as recited in Claim 1, further comprising the steps of:

receiving a plurality of software source code modules that support new features or platforms at a respective plurality of business unit pre-integration branches;

at each business unit, testing each feature of the software source code modules of that business unit individually, in combination with each other feature individually, and in combination with all other features;

committing one or more of the plurality of software source code modules from the one or more of the business unit pre-integration branches to a central pre-integration branch only when such testing is successful; and

committing the plurality of software source code modules from the central pre-integration branch to the early development branch when all the modules have been committed from the business unit pre-integration branches to the central pre-integration branches.

22. A method as recited in Claim 19, further comprising the step of generating, using the early development branch, a new early development release containing pre-tested source code for new features and platforms only when the plurality of software source code modules has been committed from the central pre-integration branch to the early development branch.

23. A computer-readable medium comprising one or more stored sequences of instructions for providing release control using early deployment releases of a software system, the early deployment releases containing support for new features and platforms, which instructions, when executed by one or more processors, cause the one or more processors to perform the steps of:

a. providing an early development branch of a software release that is designated for incorporation of support for new features and platforms;

b. receiving, from a plurality of integration units, a plurality of pre-tested source code modules, wherein each of the pre-tested source code modules comprises one or more new features or supports one or more new platforms;

c. committing the pre-tested source code for new features and platforms into the early development branch; and

d. using the early development branch, generating a new early development release containing pre-tested source code for new features and platforms.

24. A computer-readable medium as recited in Claim 23, further comprising the steps of:

receiving and testing a plurality of software source code modules that support new features or platforms at a respective plurality of business unit pre-integration branches;

committing one or more of the plurality of software source code modules from the one or more of the business unit pre-integration branches to a central pre-integration branch only when such testing is successful; and

committing the plurality of software source code modules from the central pre-integration branch to the early development branch when all the modules have been committed from the business unit pre-integration branches to the central pre-integration branches.

25. A computer-readable medium as recited in Claim 24, further comprising the step of generating, using the early development branch, a new early development release containing pre-tested source code for new features and platforms only when the plurality of software source code modules has been committed from the central pre-integration branch to the early development branch.

26. A computer-readable medium as recited in Claim 23, further comprising the steps of:

receiving a plurality of software source code modules that support new features or platforms at a respective plurality of business unit pre-integration branches;

at each business unit, testing each feature of the software source code modules of that business unit individually, in combination with each other feature individually, and in combination with all other features;

committing one or more of the plurality of software source code modules from the one or more of the business unit pre-integration branches to a central pre-integration branch only when such testing is successful; and

committing the plurality of software source code modules from the central pre-integration branch to the early development branch when all the modules have been committed from the business unit pre-integration branches to the central pre-integration branches.

27. A computer-readable medium as recited in Claim 24, further comprising the step of generating, using the early development branch, a new early development release containing pre-tested source code for new features and platforms only when the plurality of software source code modules has been committed from the central pre-integration branch to the early development branch.

28. A system as recited in Claim 8, further comprising the steps of:

receiving and testing a plurality of software source code modules that support new features or platforms at a respective plurality of business unit pre-integration branches;

committing one or more of the plurality of software source code modules from the one or more of the business unit pre-integration branches to a central pre-integration branch only when such testing is successful; and

committing the plurality of software source code modules from the central pre-integration branch to the early development branch when all the modules

have been committed from the business unit pre-integration branches to the central pre-integration branches.

29. A system as recited in Claim 28, further comprising the step of generating, using the early development branch, a new early development release containing pre-tested source code for new features and platforms only when the plurality of software source code modules has been committed from the central pre-integration branch to the early development branch.

30. A system as recited in Claim 8, further comprising the steps of:

receiving a plurality of software source code modules that support new features or platforms at a respective plurality of business unit pre-integration branches;

at each business unit, testing each feature of the software source code modules of that business unit individually, in combination with each other feature individually, and in combination with all other features;

committing one or more of the plurality of software source code modules from the one or more of the business unit pre-integration branches to a central pre-integration branch only when such testing is successful; and

committing the plurality of software source code modules from the central pre-integration branch to the early development branch when all the modules have been committed from the business unit pre-integration branches to the central pre-integration branches.

31. A system as recited in Claim 8, further comprising the step of generating, using the early development branch, a new early development release containing pre-tested source code for new features and platforms only when the plurality of software source code modules has been committed from the central pre-integration branch to the early development branch.